

IUT de VANNES
Département Informatique
BP 561
56017 VANNES CEDEX

LE RESTE THIBAULT
GROUPE C
Tuteur : M. GAFFIOT

DEVELOPPEMENT D'UNE APPLICATION POUR LA GESTION DE FICHES ETUDES

Stage effectué du 17 Avril au 23 Juin 2006 en vue de l'obtention du DUT Informatique, sous la responsabilité de :
M. BLANC Directeur de l'ADRIA SENSO.

Au sein de l'entreprise : ADRIA SENSO
29196 Quimper Cedex

SOMMAIRE

TABLE DES ILLUSTRATIONS

Remerciements

Je tiens à remercier :
Mon maître de stage, JEAN-HUGUES BLANC, pour son intégration au sein de l'entreprise ADRIA SENSO et pour son suivi du stage.
DAVID et NICOLAS pour leur aide pour l'installation su serveur et de la base de données.
MICKAEL AUCOUTURIER pour nous avoir fourni les codes sources.
Les testeurs de l'application, JESSY, MURIELLE ET ALEXIS.
ADRIEN BONNAIRE pour le travail effectué.
GILLES GAFFIOT mon tuteur de stage, pour sa visite au sein de l'entreprise.
FRANCOIS MORICE, le responsable des stages, pour m'avoir proposer ce stage.

1. INTRODUCTION

Ce stage a été effectué dans le cadre de l'obtention du DUT Informatique, de avril à juin 2006 au sein de l'entreprise ADRIA SENSO située à Quimper. Le stage a été effectué sous la direction de JEAN-HUGUES BLANC, directeur de l'ADRIA SENSO.

J'ai choisi ce stage, proposé par Mr. François MORICE, car il contenait toutes les étapes d'un développement informatique, de l'analyse jusqu'au codage d'une application. De plus l'entreprise ADRIA se situe à Quimper qui se situe à coté de mon domicile familial.

Ce stage s'est déroulé avec une équipe de 2 étudiants du département informatique de l'IUT de Vannes, moi-même et Adrien Bonnaire. Durant ce rapport je parlerais des choix, des actions effectuées mais il ne faut en aucun cas oublié que ce stage a été effectué par 2 personnes qui ont produites la même charge de travail.

Je commencerais par présenter l'entreprise du stage, puis je parlerais des objectifs du stage suivi de la réalisation du projet. Finalement, je présenterais les résultats obtenus au cours du stage.

2. L'ADRIA SENSO

2.1. Le groupe ADRIA et ses missions



Logo de l'ADRIA

L'ADRIA a été fondé en 1971 sous une forme associative. Au fur et à mesure des années, ses compétences se sont développées. Son rôle est d'accompagner les entreprises du secteur agroalimentaire dans leur processus d'innovation, dans la maîtrise de la qualité et dans la formation du personnel.

Depuis juillet 2003, l'association s'est scindée en trois entités :

- ADRIA DEVELOPPEMENT (recherche microbiologique, développement produits et procédés, formation professionnelle et conseil, réglementation)
- ADRIA LABORATOIRES (analyses microbiologiques et chimiques)
- ADRIA SENSO (évaluations sensorielles).

2.2. ADRIA SENSO au sein du groupe ADRIA



Logo de l'ADRIA SENSO

ADRIA SENSO est une société spécialisée dans l'évaluation et la caractérisation sensorielle. Elle réalise des prestations de tests consommateurs, de profils sensoriels et de la formation aux techniques sensorielles pour les clients industriels, distributeurs, groupement qualité, organismes certificateurs, fédérations, restaurateurs… sur des gammes de produits alimentaires (des biscuits au pet-food en passant par les produits surgelés, frais) et hors alimentaires tels que la cosmétique (crème, démaquillants, savon, shampoing…) ou la parapharmacie (dentifrice…). Cette société est présente à Toulouse, Quimper et Clermont Ferrand.

Aujourd'hui, de plus en plus d'industriels souhaitent connaître l'appréciation des consommateurs avant de lancer un nouveau produit ou une nouvelle recette. Cela leur permet de réduire les risques de lancement de produits et d'adapter les produits aux attentes du consommateur qui évoluent sans cesse.

ADRIA SENSO a constitué depuis plus de 10 ans un panel sans cesse grandissant de consommateurs qu'elle a qualifié et optimisé.
Ainsi, l'ensemble des segments de population est représenté dans ce panel avec une spécialisation rare sur le segment des enfants et des jeunes adolescents.
Ce panel se complète avec un large panel d'animaux domestiques.

2.3. Carte d'identité de l'ADRIA SENSO

Adresse : Adria Creac'h Gwen 29196 Quimper Cedex
Téléphone : 02.98.10.18.10
Fax : 02.98.10.24.47
Site internet : http://www.adria.tm.fr/
Secteur d'activité : agroalimentaire

La raison sociale de l'entreprise est SAS : Société par Actions Simplifiée (La société par actions simplifiée (SAS) est une forme de société commerciale définie par les articles L227-1 à L227-20 et L244-1 à L244-4 du Code du commerce français).

Son chiffre d'affaire est de 2.200.000 € par an.

Elle a de nombreux concurrents à travers toute la France, les principaux sont : Eurofins, Silliker et Alcontrol…

Quelques chiffres concernant l'entreprise :

- Une équipe de 22 personnes
- Plus de 1 200 tests par an
- Plus de 45 000 consommateurs qui ont goûté un produit
- 2 salles de dégustation à Quimper et à Toulouse de 20 et 15 boxes,
    - conforme à la norme XP V 09-500
    - informatisées
    - équipées d'une cuisine« type cuisine collective »(matériel de cuisson, plaques, four, frigo positif et négatif)

2.4. L'informatique au sein du groupe ADRIA SENSO

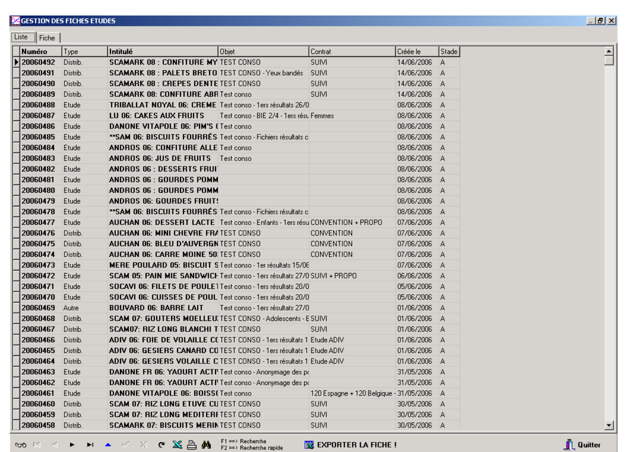
Le groupe ADRIA SENSO ne possède pas de service informatique. Nous avons donc représenté le service informatique lors de ces 10 semaines de stage.
Le groupe ADRIA à Quimper comporte 3 informaticiens qui nous on aidé au début du stage.

3. OBJECTIFS DE LA MISSION TECHNIQUE

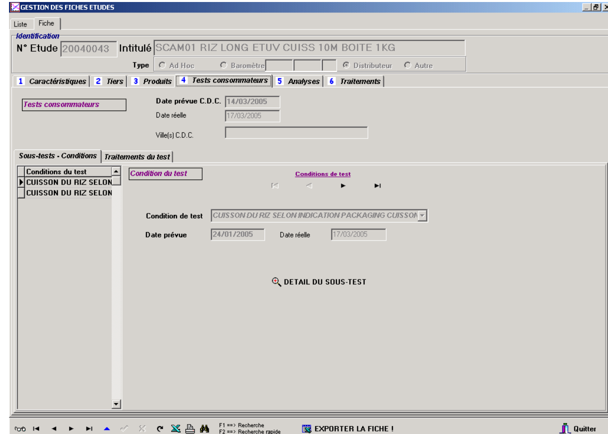
3.1. Sujet du stage

Pour réaliser ses tests consommateurs et analyses, le groupe ADRIA SENSO utilise des fiches études. Celles-ci contiennent les informations nécessaires à la gestion des tests consommateurs.

Actuellement, ces fiches sont gérées par le logiciel *SIGES* qui doit être installé sur chaque machine. Ce logiciel est programmé en Delphi. Il a été développé par l'entreprise *Team Partner* qui n'existe plus aujourd'hui. Il n'y a donc plus de maintenance effectuée.
Ce logiciel demande beaucoup de ressources et monopolise une grande partie de la bande passante pour les échanges de données entre les différents pôles de l'ADRIA SENSO. Il utilise une ligne SDSL (Symetric Digital Subscriber Line) qui coûte très cher à l'entreprise.



La liste des études de SIGES



La page des test consommateurs de SIGES

Mon stage consiste à développer un outil plus rapide, plus pratique et économique que le logiciel déjà existant.

Ce nouvel outil sera une application Web, de type Intranet.

3.2. Le cahier des charges fonctionnel

3.2.1. Présentation globale du projet et contraintes

- **Les utilisateurs**

Ce sont les employés d'ADRIA SENSO, exemple : les chargés de Panel (CP)
Ils sont répartis entre Quimper et Toulouse.
Il y a des droits différents en fonction de la hiérarchie des employés, exemple : l'administrateur a tous les droits.

- **Contraintes techniques**

L'application développée devra :

  - être compatible avec Windows 2000 Professionnel NT et supérieur.
  - fonctionner sous Internet Explorer 6 et supérieur.
  - utiliser une base de données Oracle déjà existante.

Une étude devra être réalisée pour connaître la meilleure méthode de développement de l'application (langage de programmation) ; exemple : PHP5, ASP.NET …
Elle prendra en compte les contraintes techniques de l'entreprise.

- **Les documents disponibles**

Les documents suivants sont disponibles :

  - Le document d'analyse (version 1) de l'application LEONI
  - Le manuel utilisateur (version 1) de l'application LEONI V2
  - Le document de description des tables de la base de données ainsi que les modèles conceptuels attachés de l'application LEONI.

- **La base de données Oracle**

Actuellement, une base de données Oracle est disponible. Elle contient 21 tables telles que : Analyses, Produits, Familles … C'est une base très lourde, par exemple, la table Produits contient environ 15000 tuples (lignes).
Cette base ne devra pas être modifiée. Pour les tests de l'application, il faudra réaliser une copie de la base.

NB : Pour l'instant la base de données contient des enregistrements inutiles. Ainsi, il pourra être possible de créer un script pour les supprimer.

3.2.2. Principales fonctionnalités attendues

- **Liste des études**

Cette rubrique permettra d'afficher la liste de toutes les fiches études.
La liste fournira les principales informations de chaque étude pour faciliter les recherches : son numéro, son type (étude, baromètre, autre …), son intitulé, son objet, son contrat, sa date de création et son stade (A, C …).
Il devra être possible de sélectionner une fiche pour en avoir ses informations ou pour la modifier. Il est aussi possible de créer une nouvelle fiche étude.

- **La fiche étude**

La fiche étude sera composée de cinq fonctions principales :

- Caractéristiques générales

Contient les caractéristiques générales d'une étude.

- Tiers

Permet d'ajouter ou de supprimer un ou plusieurs tiers de l'étude.
Contient la liste des tiers demandeurs de l'étude.

- Produits

Permet d'ajouter ou de supprimer un ou plusieurs produits de l'étude.
Contient la liste des produits de l'étude.

- Tests consommateurs

Permet de créer, modifier ou supprimer des tests consommateurs pour une étude.
Contient la liste des tests consommateurs de l'étude.

- Analyses

Permet d'ajouter, modifier ou supprimer une analyse pour l'étude en cours.
Contient la liste des analyses pour l'étude en cours.

- **Recherche et impression**

A tout moment, l'utilisateur pourra imprimer la fiche en cours. Il pourra aussi imprimer la liste des fiches études. De même, il pourra effectuer une recherche sur celles-ci. Elle pourra être faite en fonction de différentes requêtes (délais, typologie, date…). Par la suite l'utilisateur aura la possibilité d'imprimer les résultats de cette recherche.

- **Interface**

L'application devra ressembler à l'application déjà existante (SIGES). Néanmoins, des améliorations graphiques pourront être développées.
Par exemple l'interface pourra comporter des onglets ou des menus qui correspondront aux fonctions de la fiche étude.
On optera pour une interface simple et modulable.
L'utilisateur devra facilement savoir où il se trouve dans l'application et ne devra pas se perdre dans les différents menus.

3.2.3. Fonctionnalités annexes

- **Configuration de l'application**

Des modules de configuration pourront être développés. Exemple : Attribution des droits des utilisateurs, configuration de l'interface …

- **Menu des actions**

Ce menu est affiché de nombreuses fois sur les différentes fenêtres de l'application existante. Ce menu permet l'ajout, la modification, la suppression de différentes options contenues dans les tables de la base de données. Exemple : Souscripteurs, Test consommateurs, Tiers demandeurs…
Ce menu devra être remanié et simplifié pour plus d'ergonomie et de lisibilité.

3.2.4. Mise en oeuvre

- **Installation**

Un fichier README sera présent indiquant les dernières informations sur l'installation de l'application.

- **Formation**

Un guide d'utilisation et de développement sera fourni, décrivant les fonctionnalités de l'application en fonction du type de profil.

- **Maintenance**

Les développeurs ne fourniront pas de nouvelles versions de la solution et n'assureront pas d'assistance technique.
Les tests seront réalisés par les développeurs de l'application. Ils seront fournis dans une annexe.
Un fichier BUGS contiendra la liste des bugs de l'application.
Un fichier TODO contiendra la liste des évolutions futures de l'application.

3.3. Moyens mis en œuvre à disposition du stagiaire

Le travail s'effectue en équipe de deux étudiants. Ce binôme est composé de moi-même et Adrien Bonnaire de l'IUT de Vannes. Le travail en binôme est un avantage car nous avons déjà l'habitude de travailler par deux dans le cadre des projets et des travaux personnels de l'IUT.

Nous disposons de 3 machines :

- 2 ordinateurs de développement équipés de Windows 2000 NT professionnel.
- 1 serveur équipé de Windows 2000 serveur NT

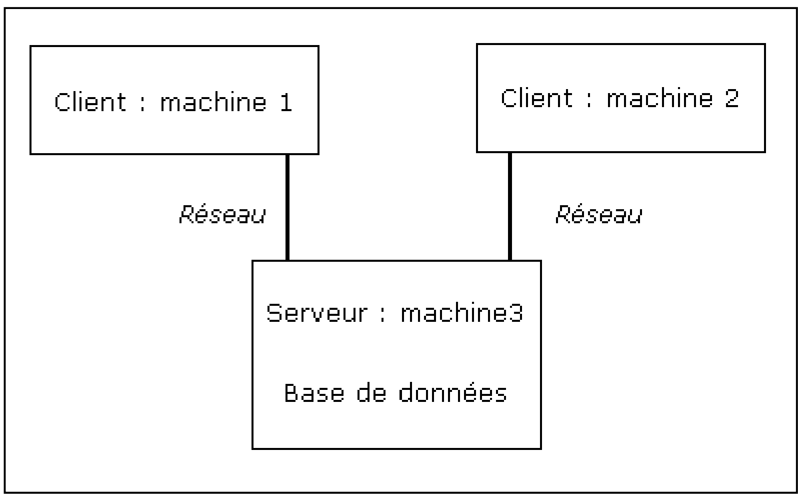


Schéma de la relation entre les machines de travail

Nous sommes aidé par les deux informaticiens de l'ADRIA Quimper : David et Nicolas.

3.4. Planification du projet

Le planning prévisionnel est le suivant :

- **2 Semaines d'analyse et de mise en place de la base de test**

- Conception du cahier des charges.
- Conception du dossier d'analyse des langages serveur.
- Conception du dossier d'analyse et conception.
- Mise en place de la base de test.

- **5 Semaines de codage**

- Mise en place de la base de données de test.
- Développement des pages HTML et du CSS.
- Développement des pages PHP.

- **3 Semaines de tests et finitions**

- Test des bugs de l'application.
- Les futurs utilisateurs testeront l'application pour faire part de leurs remarques.
- Déploiement de l'application à Toulouse pour tester son bon fonctionnement.
- Création du manuel d'utilisation.

4. REALISATION DU PROJET

4.1. Les outils et langages utilisés

Les outils utilisés seront des outils libres et gratuits. En effet comme l'entreprise est dans une optique commerciale, il faut faire très attention aux licences des logiciels utilisés. Par exemple, certain logiciels utilisés à l'IUT comme « Poséidon » on une licence de type « Not for commercial use ». Néanmoins l'entreprise a en sa possession plusieurs licences payantes que nous pourront utiliser. Les postes sont sous Windows 2000 Professionnel.

4.1.1. Les langages utilisés

PHP5
Le langage serveur utilisé est le PHP5. En effet le PHP5 est orienté objet, il est gratuit et dispose d'une grande communauté sur Internet. (cf : Dossier d'analyse des langages serveurs). Il permet l'affichage et la modification des données.

HTML & CSS
Pour le développement de l'interface l'application nous utilisons le HTML et le CSS qui sont les langages utilisés pour les interfaces des sites Web.

Oracle
Pour le système de gestion de base de données, nous n'avons pas eu le choix. La base existante étant déjà sous Oracle.

XML
C'est le langage de balisage utilisé pour la gestion des collaborateurs.

Javascript
Permet d'afficher un popup pour la vérification de l'enregistrement des données. S'exécute sur le poste client.

4.1.2. Les logiciels utilisés

- **Bouml version 2.14.1** (http://bouml.free.fr/ | Gratuit)
Logiciel permettant la création de diagrammes de cas d'utilisations, de séquences, synoptiques de Jacobson…

- **Siges version 3.1.0.8** (licence acquise par l'ADRIA)
C'est le logiciel actuellement utilisé par l'ADRIA SENSO pour la gestion de ses fiches études.

- **Oracle version 8i** (licence acquise par l'ADRIA)
Logiciel permettant la gestion de la base de données Oracle.

- **ConTEXT version 0.98.3.1050** (http://www.context.cx/ | Gratuit)
Logiciel permettant la coloration syntaxique de l'HTML et du CSS.

- **Client Terminal Server** (licence acquise par l'ADRIA)
Logiciel permettant d'avoir une vue sur le serveur de la base de donnée.

- **SQLPLUS**
Logiciel permettant d'effectuer des requêtes sur les bases de données.

4.2. Conception générale

4.2.1. L'architecture de l'application existante : 2 bases de données

L'application existante (*SIGES*) utilise 2 bases de données, l'une située à Quimper, et l'autre située à Toulouse. Les différentes modifications effectuées à l'aide du logiciel doivent être réalisées sur les 2 bases de données. Pour cela, l'entreprise utilise une ligne SDSL : Technologie de transmission où les transferts vont aussi vite en montant qu'en descendant, contrairement à l'ADSL. (*SDSL: Symmetric Digital Subscriber Line)*

Cette ligne permet de faire circuler les nombreuses requêtes pour mettre à jour chaque base de données.

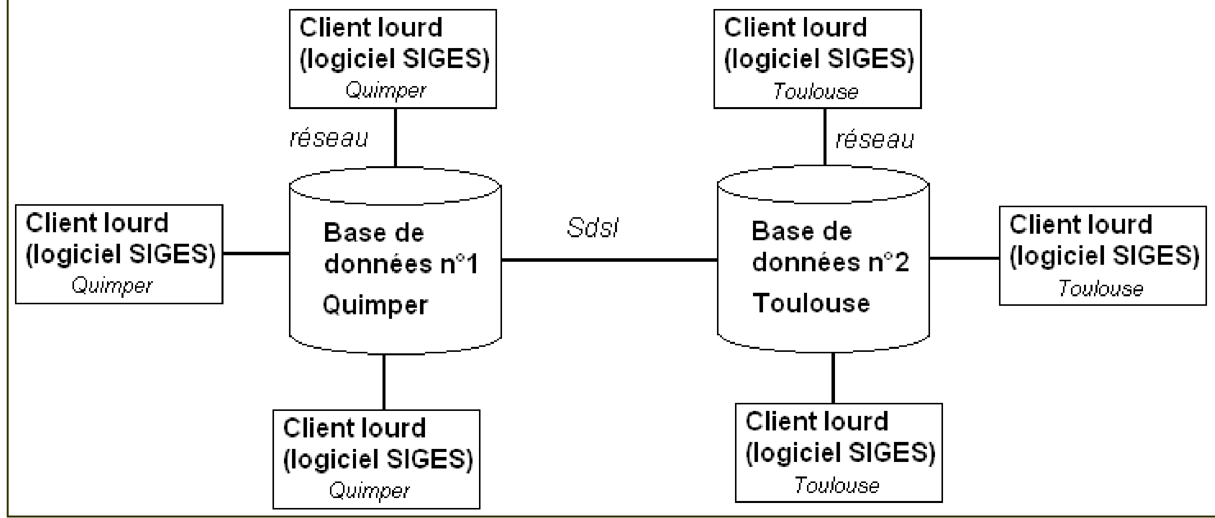


Schéma de l'architecture de l'application SIGES

4.2.2. L'architecture de la nouvelle application : 1 base de données

Le schéma suivant illustre la nouvelle architecture de l'application. L'avantage de cette architecture et qu'elle utilise une base de données unique. Ainsi la base de données et située uniquement à Quimper. Chaque utilisateur peut accéder à l'application à l'aide de son navigateur web. Une ligne ADSL suffit donc ici.

Le schéma suivant illustre l'architecture globale de l'application.

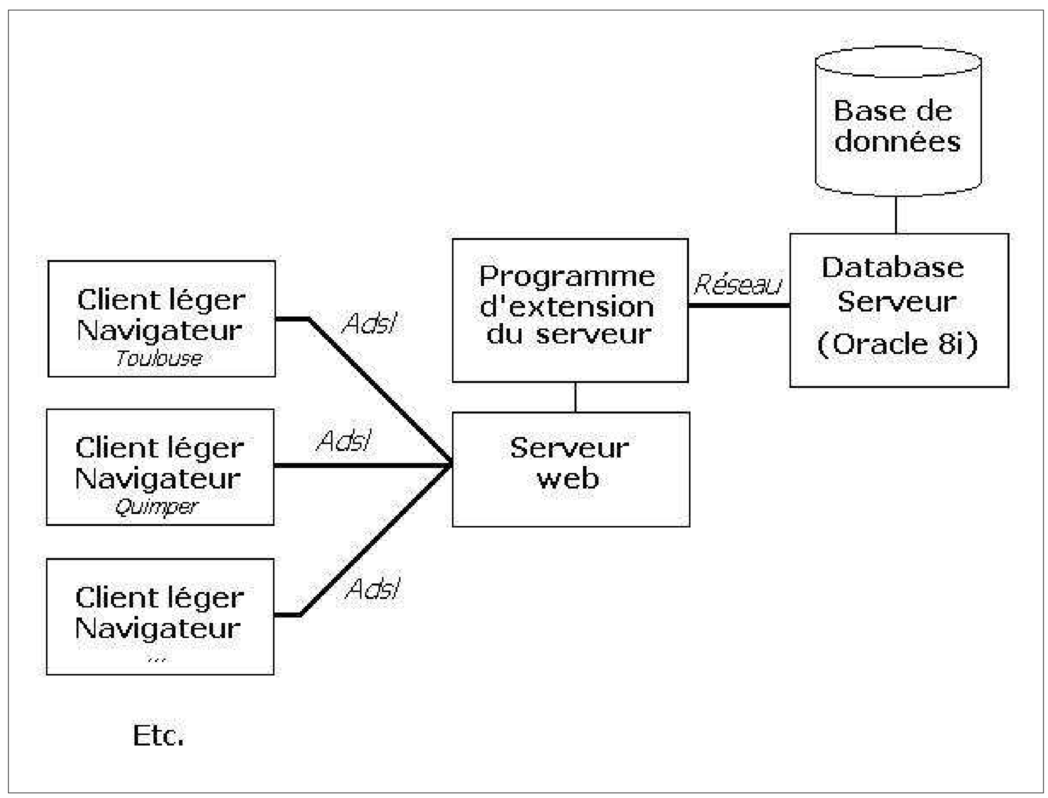


Schéma de l'architecture de la nouvelle application

C'est une **architecture 3-Tiers**, celle-ci est composée de 3 couches :

- La couche présentation (ou affichage) associée au client qui de fait est dit "léger" dans la mesure où il n'assume aucune fonction de traitement.
- La couche fonctionnelle liée au serveur, qui dans de nombreux cas est un serveur Web muni d'extensions.
- La couche de données liée au serveur de base de données (SGBD).

4.3. Conception détaillée

Après différentes recherches sur les architectures des applications Web. Nous avions tout d'abord pensé à utiliser un Framework, c'est à dire un logiciel qui permet de structurer les applications PHP afin de les rendre plus facilement maintenable et évolutive. Nous avions pensé au Framework *SMARTY* (http://smarty.php.net) qui semblait simple d'utilisation et efficace. Finalement, à la suite de la découverte d'un article sur developpez.com, nous avons décidé de développer notre propre système de template.

Voici l'organisation interne de l'application.

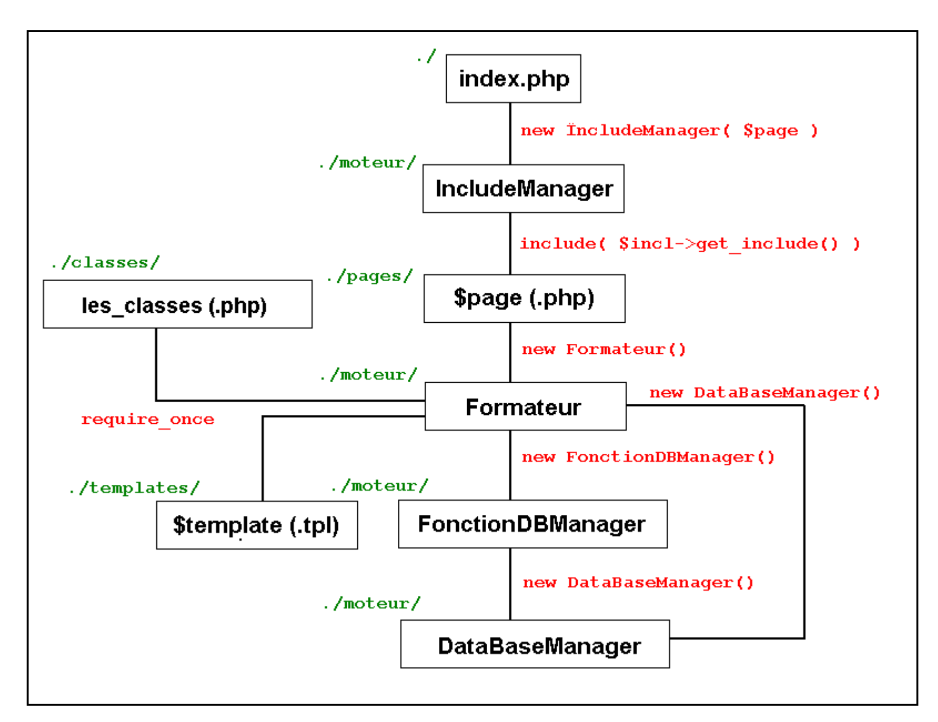


Schéma de l'architecture détaillée de l'application Web

Le fichier index.php est le fichier que l'on appelle pour lancer l'application.

L'IncludeManager est une classe permettant de définir quelle page à inclure (à appeler) en fonction des liens et onglets cliqués.

Le Formateur est une classe permettant de formater chaque partie de page, chaque tableau, liste déroulante… au format HTML. Il utilise la plupart du temps la classe FonctionDBManager car il formate le plus souvent des données de la base de données. En effet, la classe FonctionDBManager permet de créer une connexion à la base de données et d'effectuer des requêtes sur celle-ci.
Le Formateur utilise des template qui sont des parties de chaque page (formulaire, bannière) au format HTML.

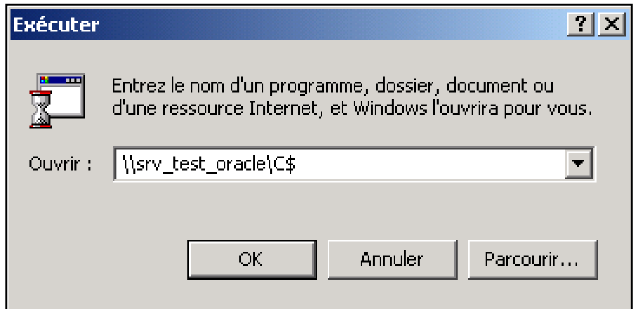
4.4. La mise en place de la base de données de test

Avant de débuter le codage de l'application, nous avons mis en place une base de données de test. C'est une copie de la base de données qui est utilisée par le logiciel *SIGES*. En effet, nous n'avons pas directement développé l'application autour de la vraie base de donnée pour ne pas modifier celle-ci.
La création de la base de données de test a été un gros problème.

Tout d'abord, nous avons créé une base de données vide a l'aide d'*Oracle 8i*. Dans un premier temps, cette création n'a pas été possible sur les postes mis à notre disposition. (*Windows 2000 professionnel version NT*). En effet une erreur (de Dll) arrêtait la création de la base à chaque fois que nous lancions la création d'une base vide. Après différents jours de recherche, Nicolas a installé un autre poste équipé de *Windows serveur*. Il a ensuite créé une base vide sur ce poste, la création a fonctionné sans erreur.

Ensuite, nous avons généré les tables de la base de test et les remplir à l'aide d'un dump fourni par David. Cette partie a posé d'autres problèmes. En effet la base de test ne se remplissait pas car l'utilisateur 'AGID' n'existait pas. Nous avons donc créé cet utilisateur à l'aide du logiciel *SQLPLUS*. Ensuite nous avons mis les droits DBA (Data Base Administrator) à cet utilisateur pour avoir tous les droits sur la base de Test (SELECT, UPDATE…). Ensuite, nous avons généré les tables et leur remplissage. Cela a fonctionné avec seulement quelques avertissements.

La base de données de test est donc maintenant présente sur le nouveau poste serveur, accessible par le réseau de l'entreprise et de nom : *srv_test_oracle*.



Commande pour accéder au C:/ du serveur de test

Le serveur est une machine ancienne : Pentium 3 500Mhz auquel des barrettes de mémoire ont été ajoutées pour monter la mémoire (RAM) à 512mo. Le disque dur est un disque rapide tournant à 10000 tours/minutes.
Bien que la machine soit ancienne, la plupart des requêtes simples mettent moins d'une seconde à êtres exécutées.

4.5. La mise en place des outils nécessaires au fonctionnement de l'application

4.5.1. Le serveur Apache



Le logo d'Apache

Après avoir installé la base de données de test, nous avons installé sur la machine serveur un serveur Apache pour afficher les différentes pages de l'application.

Le processus d'installation est décrit dans le document de développement.

Après cette installation, nous avons configuré le serveur apache :
la configuration du serveur Apache consiste en la modification de certaines valeurs du fichier de configuration d'apache, le fichier *http.conf*.

La configuration du serveur est aussi décrite dans le document de développement.

Par exemple, le « DocumentRoot » est une variable qui spécifie le répertoire racine du site. Seules les pages présentes dans ce dossier sont interprétées par le serveur.
Pour modifier le DocumentRoot, nous avons changé la ligne du fichier httpd.conf :
« *DocumentRoot 'c:/Program Files/Apache Group/Apache2/htdocs'* »
par la ligne suivante :
« *DocumentRoot 'c:/www/AdriaSite/'* ».

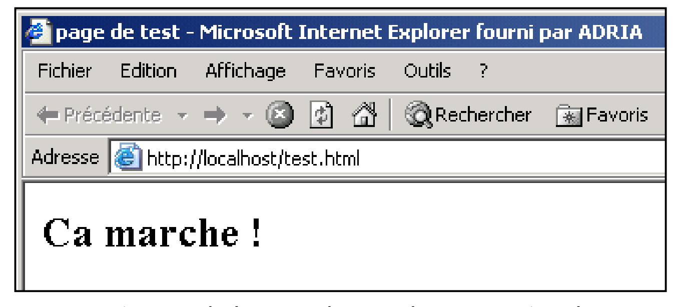
Pour tester le bon fonctionnement du serveur Apache, nous avons créé une page nommée test.html et nous l'avons placé dans le dossier '*c:\www\AdriaSite*'.

Voici, le code de la page de test :

```
<html>
<head>
      <title>page de test</title>
</head>

<body>
      <div style="font-size:24px;font-weight:bold;">
          Ca marche !
      </div>
</body>
</html>
```

Ensuite, nous avons lancé le serveur Apache et tapé l'adresse *http://localhost/test.html* dans notre navigateur (Internet Explorer). L'affichage du message « Ca marche ! » nous a montré que le serveur Apache fonctionne correctement.



Aperçu de la page de test du serveur Apache

4.5.2. PHP5 et sa liaison avec le serveur Apache

Une fois que l'installation du serveur Apache été terminée, nous avons installé PHP5. Pour configuré PHP5, nous avons modifié le fichier '*php.ini*' (cf : Document de développement).

Nous avons créé une page « *essai.php* » pour tester le fonctionnement de PHP5.

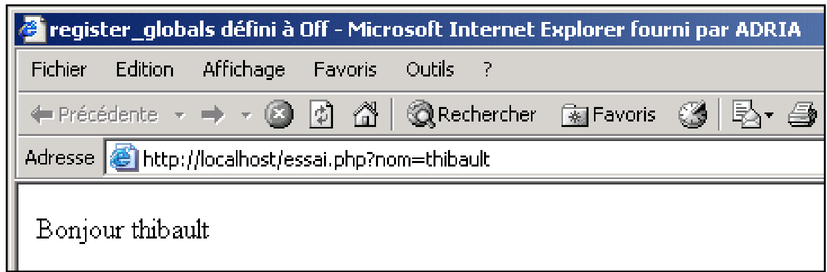
Voici le code du fichier essai.php :

```
<html>
<head>
<title>register_globals défini à Off</title>
</head>

<body>
<?php
          echo 'Bonjour '.$_GET['nom'];
?>
</body>
</html>
```

Pour pouvoir tester cette page, nous avons lié le serveur Apache avec PHP5. Le déroulement de la liaison est décrit dans le document de développement.

Ensuite, nous avons tapé l'adresse : *http://localhost/essai.php?nom=thibault* notre navigateur (Internet Explorer). L'affichage du message « Bonjour thibault » nous a montré que PHP5 et sa liaison avec le serveur Apache fonctionné correctement.



Aperçu de la page de test de PHP5

4.5.3. La liaison entre la base de données de test et le serveur Apache

La dernière étape de la mise en place des outils a été de lier le serveur Apache (équipé de PHP5) avec la base de données de test. Ce processus n'a pas posé de problème, il est expliqué dans le dossier de développement.

4.6. La préparation du codage

Avant de réellement coder l'application, nous avons réalisé différents travaux préparatoires.

4.6.1. Création de l'architecture interne de l'application

Cette étape consiste en la création des différents dossiers de l'application.

```
> Nom de l'application
  index.php
  > classes
  > conf
  > css
  > images
  > moteur
  > objets
  > pages
  > scripts
  > templates
```

Le fichier index.php est le lanceur de l'application.

Description des dossiers de l'application :

- **classes :** Contient les fichiers « classes » destinés à la programmation orientée objet, chaque classe correspond à une table de la base de données.
  Exemple : ETUDES, COLLABORATEURS, TIERS…..

- **conf :** Contient les fichiers de configuration de l'application (.xml et config.inc.php).

- **css :** Contient les feuilles de styles .css pour la mise en page des différents designs de l'application.

- **images :** Contient toutes les images de l'application.

- **moteur :** Contient les classes Php qui contiennent les fonctions nécessaires à son fonctionnement (FonctionDBManager, Formateur…).

- **objets :** Contient les objets divers (.php) qui ne correspondent pas à une table de la base de données.

- **pages :** Contient toutes les pages (accueil, liste des fiches…) du site.

- **scripts :** Contient le ou les fichiers JavaScript (.js).

- **templates :** Contient les templates (fichiers .tpl) de l'application. Ces templates sont les patrons des pages de l'application.

4.6.2. Le diagramme synoptique de Jacobson

Avant de démarrer le codage de l'application, nous avons fait une analyse poussée des besoins de des utilisateurs. Pour cela, nous avons commencé par créer un diagramme contenant les principales actions que l'utilisateur pourrait faire avec l'application et la relation qu'ont ces actions avec les acteurs de l'application :

- **L'Utilisateur**

C'est l'utilisateur de l'application. Il s'agit des employés de L'ADRIA SENSO qui ont besoin de compléter ou étudier des fiches études pour effectuer les terrains d'étude.

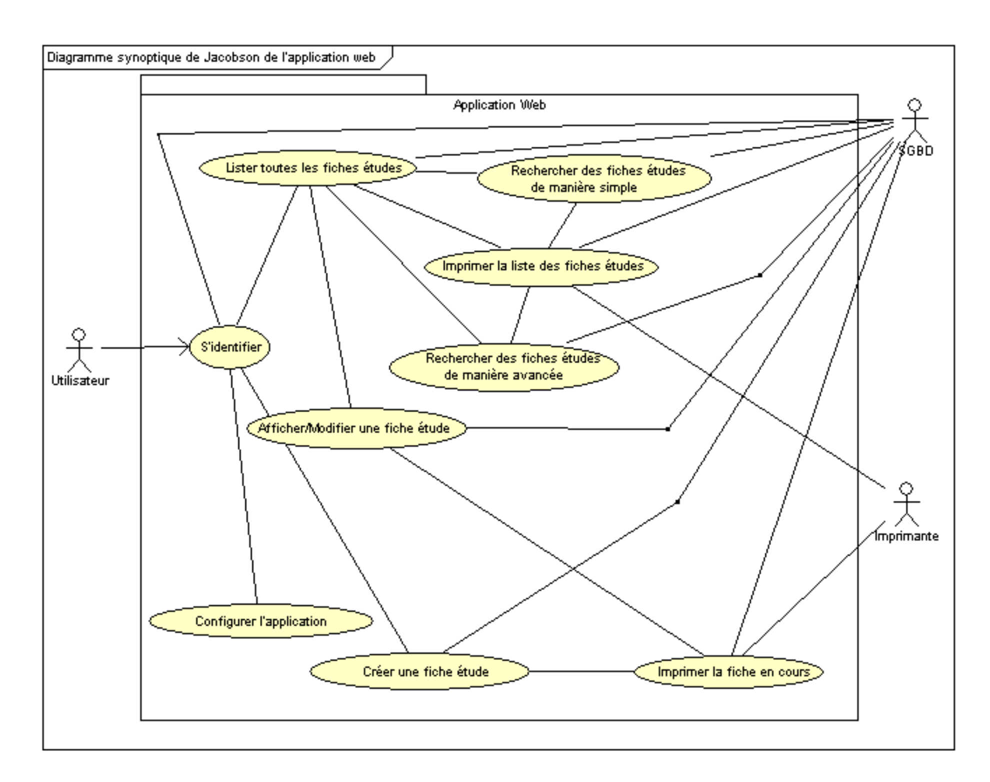
- **L'imprimante**

Elle va permettre l'impression de certains documents, par exemple les fiches études.

- **Le SGBD**

Il s'agit de la base de données Oracle où toutes les informations seront stockée, par exemple les fiches études, les utilisateurs…

Ce diagramme s'appelle le diagramme synoptique de Jacobson :



Le diagramme synoptique de Jacobson

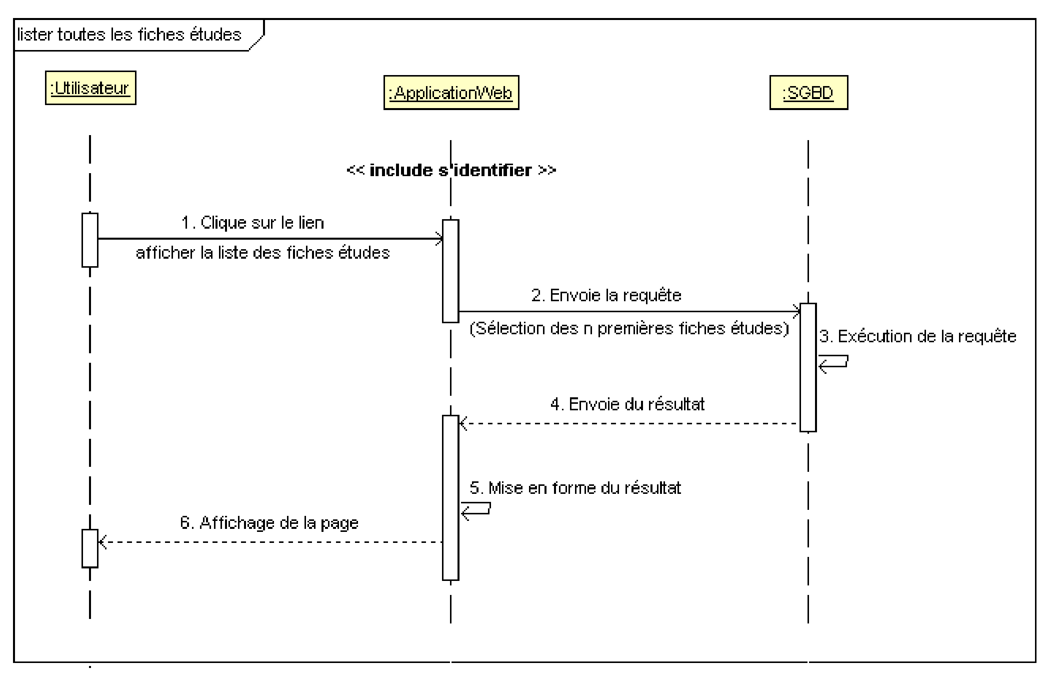
4.6.3. Les cas d'utilisation

Nous avons ensuite développé chaque cas d'utilisation, sous forme de diagramme de séquence. Chaque diagramme de séquence comprend une forme graphique et une forme résumée.

Voici l'exemple de diagramme de séquence du cas d'utilisation lister toutes les fiches études :

Forme résumée :
L'utilisateur a la possibilité d'afficher la liste de toutes les fiches études.
Pour cela, il clique sur le lien prévu à cet effet. Les n premières fiches seront affichées en fonction de leur date (les plus récentes d'abord). Par la suite, l'utilisateur pourra afficher les fiches suivantes ou précédentes en cliquant sur le lien « suivant » ou le lien « précèdent ».

Forme Graphique :



Forme graphique du cas d'utilisation « lister toutes les fiches études »

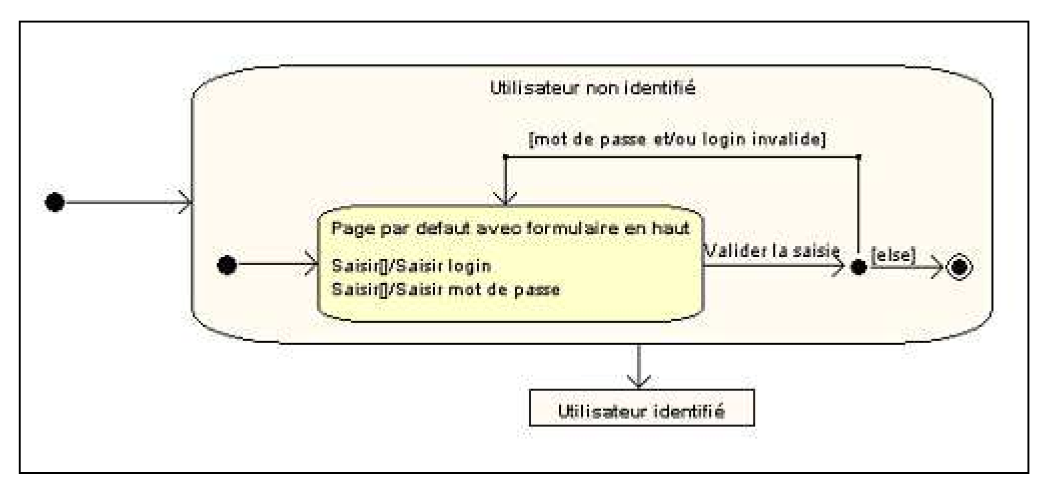
Scénarios de remplacement :
5. Si la base ne retourne aucun résultat, un message est affiché dans la page générée.

Cette analyse a permit de bien clarifier les différentes actions et tous les différents cas possibles pour ne rien oublier lors de la période de codage et pour gérer tout les cas d'erreurs.

4.6.4. Le diagramme d'Etat-transition

Finalement, nous avons créé un diagramme permettant de voir les liens qu'ont les différentes pages entre elles. Ce diagramme permet aussi de voir le cheminement possible d'un utilisateur. Si il clique sur un lien vers quelle page il sera dirigé, quelles nouvelle options s'offriront à lui..

Voici un morceau simplifié de ce diagramme correspondant à la partie connexion :



Partie simplifiée de la connexion du diagramme Etat-transition

Au départ l'utilisateur est dans l'état « utilisateur non identifié ». Ensuite il a la possibilité de saisir son login et son mot de passe et de valider sa saisie. Si le mot de passe et/ou le login son invalides, il revient au point de départ. Sinon, il se connecte et passe dans l'état « utilisateur identifié ».

4.7. Le développement de l'interface de l'application

4.7.1. Les IHMs

Tout d'abord, nous avons créé des IHM (Interface Homme Machine) pour avoir un aperçu global de l'interface de l'application. Ce sont des schémas simplifiés des différentes pages de l'application.

Voici, par exemple, l'IHM de la page d'accueil :

Logo de l'ADRIA SENSO

Login :

Mot de passe :

OK

Accueil

Ici du texte (présentation du logiciel ...)

Etc.

Pied de page

IHM de la page d'accueil

Il y a 3 parties principales :

- La bannière en haut contenant le logo de l'ADRIA SENSO et la partie log.
- La partie centrale qui contient la présentation de l'application.
- Le pied de page contenant des informations sur l'application. (programmeurs et date de création).

La bannière et le pied de page, sont des parties récurrentes de l'application, elles apparaissent dans chaque page.

4.7.2. Les pages de l'application

- **Le choix du langage :**

Nous avons ensuite développé en HTML et CSS les différentes pages de l'application.

Qu'est ce qu'est le HMTL et CSS ? :

Le CSS est utilisé pour définir les couleurs, les polices, le rendu, et d'autres caractéristiques liées à la présentation d'un document. L'objectif est de bien séparer la structure (écrite en HTML ou similaire) et la présentation (en CSS) du document. Cette séparation permet d'améliorer l'accessibilité, de changer plus facilement de structure et de présentation, et de réduire la complexité de l'architecture d'un document.

HTML ne décrit que l'architecture interne, et CSS décrit tous les aspects de la présentation. CSS peut définir couleur, police, alignement de texte, taille, position…

Ainsi, les avantages des feuilles de style (CSS) sont multiples :

- La structure du document et la présentation sont gérés dans des fichiers séparés.
- La conception d'un document se fait dans un premier temps sans se soucier de la présentation, ce qui permet d'être plus efficace.
- Un même document peut donner le choix entre plusieurs feuilles de style (impression ou lecture à l'écran par exemple). Certains navigateurs Web permettent d'accéder facilement à un choix de feuilles de style.
- Le code HTML est considérablement réduit en taille et en complexité, puisqu'il ne contient plus de balises de présentation.

CSS utilise de nombreux mots-clés anglais destinés à caractériser les éléments HTML. Voici l'exemple d'une portion de feuille de style :

```
p { font-size: 110%;
    font-family: Verdana, sans-serif;
}

h1 { color: white;
     background: red;
}
```

Ce code CSS définit l'élément p (paragraphe) avec une taille de 110% et une police Verdana. Quant aux titres (éléments h1) ils seront en blanc, sur fond rouge.

- **La mise en place des pages**

Dans un premier temps, chaque page a été créé sans les formulaires. C'est-à-dire que la partie centrale été vide pour chaque page.



Aperçu de l'application lors du développement

Nous avons créé chaque page en la construisant bloc par bloc.

Ici, il y a différents blocs :

- La bannière
- La fenêtre centrale qui contient
  - Un bandeau d'identification
  - Les onglets
  - Le formulaire (ici pas implémenté)
- Le pied de page

Chaque bloc est en fait un Template. Un Template est un patron, c'est un fichier contenant du code HTML. Ainsi chaque page est facilement modulable. C'est à dire qu'on peut très facilement enlevé la bannière par exemple, ou généré chaque bloc indépendamment des autres.

4.7.3. Les formulaires

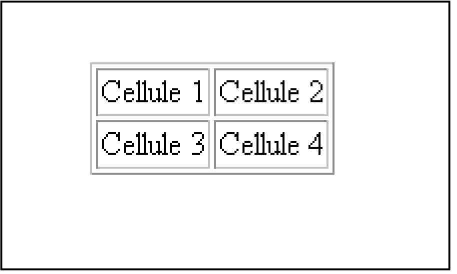
Les formulaires ont été la dernière grande étape de création de l'interface. Nous avons créé chaque formulaire sur la base des formulaires du logiciel SIGES. Ils ont été créé en HTML en utilisant les tableaux et donc sans utilisé le positionnement des feuilles de styles CSS. (Utilisation des divs)

Exemple :

Le code suivant :

```
<TABLE border="1">
       <TR>
              <TD> Cellule 1 </td>
              <TD> Cellule 2 </td>
       </TR>
       <TR>
              <TD> Cellule 3 </td>
              <TD> Cellule 4 </td>
       </TR>
</TABLE>
```

Donne le tableau suivant :



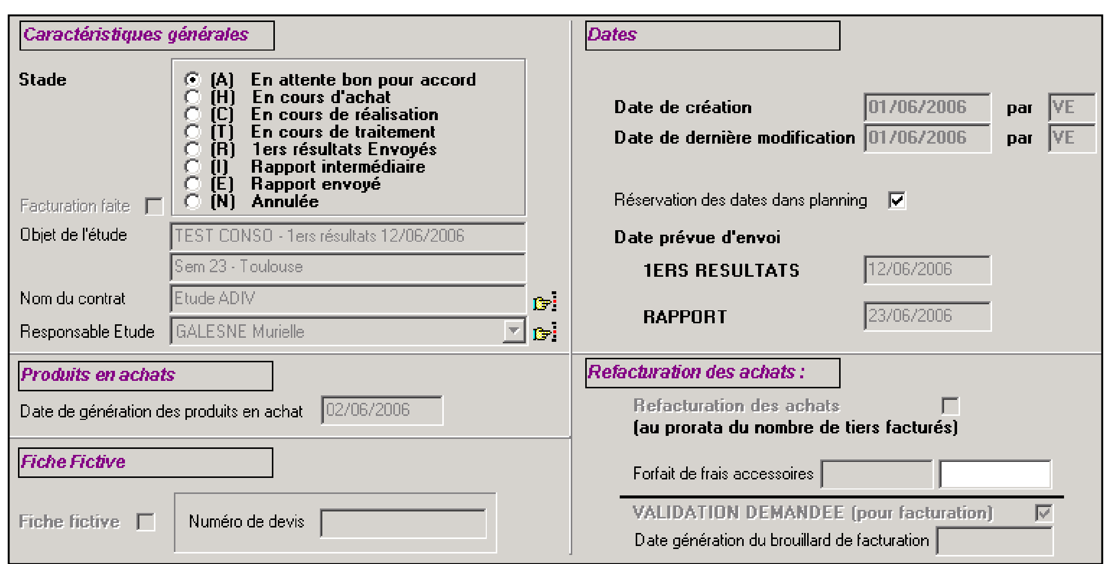
Cellule 1	Cellule 2
Cellule 3	Cellule 4

Exemple de tableau en HTML

Ainsi en jouant sur les tailles des cellules, en mettant les bordures en invisibles, en fusionnant les cellules… nous avons réussi a reproduire les formulaires de *SIGES*.

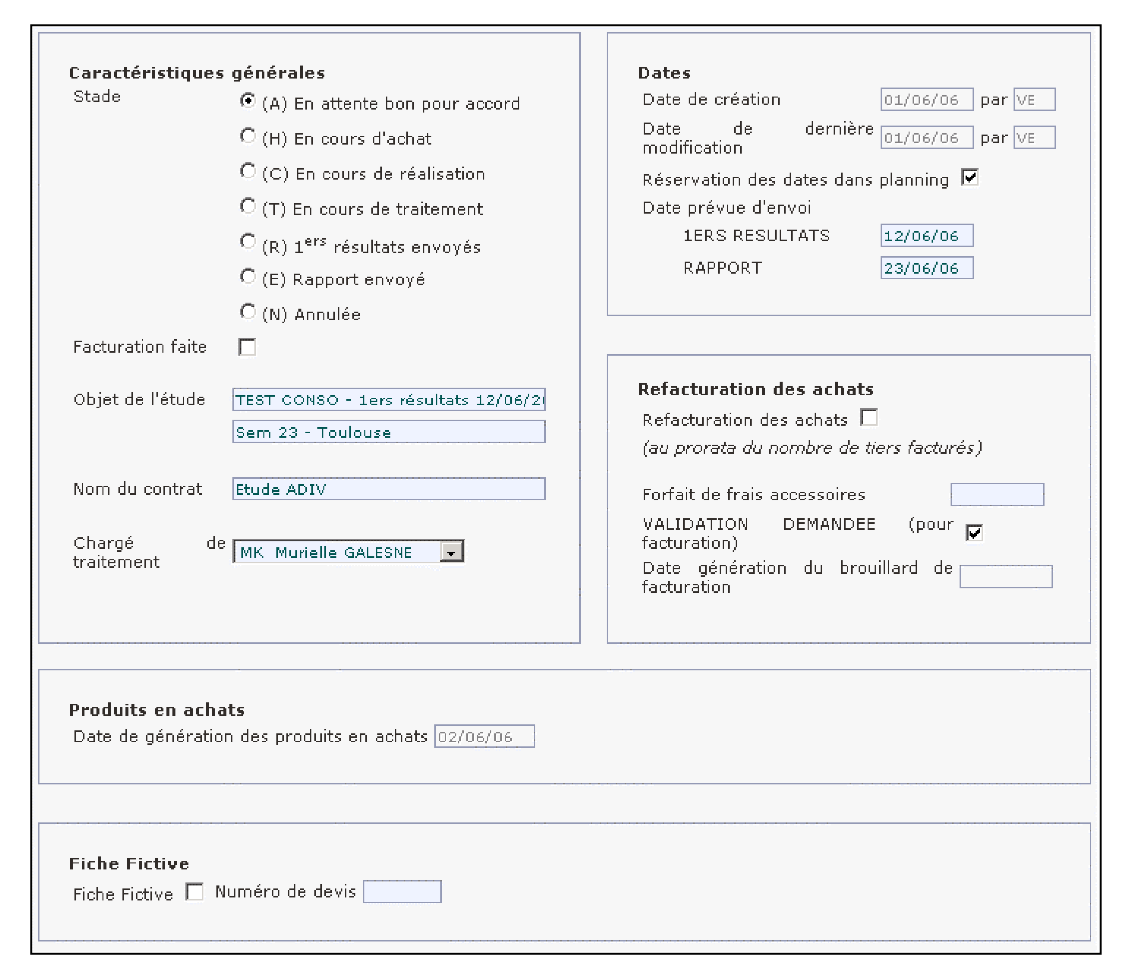
Les champs de chaque page restent les mêmes et leur disposition globale reste semblable à leur disposition sous *SIGES*.

Voici le formulaire des caractéristiques des fiches études du logiciel SIGES :



Page des caractéristiques de l'application SIGES

Voici maintenant le formulaire correspondant sous l'application Web :



Page des caractéristiques de l'application Web

4.8. Le développement du cœur de l'application

4.8.1. L'analyse des champs de SIGES

Tout d'abord, nous avons retrouvé chaque donnée de chaque champ des formulaires à l'aide du code source de *SIGES* et des diagrammes de base de données de *SIGES*. Nous avons pris des captures d'écran des différentes pages du logiciel *SIGES* auxquelles nous avons affecté pour chaque champ un numéro et la valeur associé de la base de données.

Exemple pour le bandeau « identification » d'une fiche étude :



Bandeau des fiches études de SIGES

```
1) $_GET('numFiche')
2) ETUDES.ET_INTITULE_LONG
3) ETUDES.ET_TYPE
Résultats possibles:  E = Ad Hoc
                      B = Baromètre
                      D = Distributeur
                      A = Autre
```

Pour le 1) : $_GET('numFiche') signifie qu'on récupère la variable 'numFiche' passée dans la barre d'adresse.

Pour le 2) : ETUDES.ET_INTITULE_LONG signifie qu'on récupère la variable ET_INTITULE_LONG de la table ETUDES et correspondant à la numFiche sélectionnée en 1).

Pour le 3) : Les différents résultats dépendent des boutons cochés, ici ce sont des boutons radio, ce qui signifie qu'un seul bouton peut être actif à la fois.

Cette partie du développement a été très longue et a durée plus 2 semaines. En effet, nous avons commencé à retrouver les valeurs de l'application sans les sources de *SIGES*. Nous avons eu les sources de *SIGES* plus tard mais elles concernaient une version antérieure à la version actuelle. Des tables de la base de données n'étaient pas utilisées dans ces sources et n'étaient pas non plus utilisées dans les diagrammes à ma disposition. Nous avons donc effectué de nombreuses requêtes et usé de logique pour retrouver toutes les valeurs.

4.8.2. L'analyse des fiches études

La deuxième partie de l'analyse a été de trouver des fiches études types. C'est-à-dire des fiches contenant beaucoup d'informations ou au contraire pas du tout. Ceci afin de pouvoir tester des fiches aux deux extrêmes.

Par exemple, sélection d'une étude ayant un nombre de produits supérieur ou égal à 13 :

```
SQL> SELECT PETUDES.ET_NUMERO, count(PETUDES.PR_CODE) FROM PETUDES, ETUDES
  2  WHERE PETUDES.ET_NUMERO = ETUDES.ET_NUMERO
  3  AND ETUDES.ET_ARCHIVE IS NULL
  4  GROUP BY PETUDES.ET_NUMERO
  5  HAVING COUNT(*) BETWEEN 13 AND 100;

ET_NUMERO COUNT(PETUDES.PR_CODE)
---------  ----------------------
20050196                    13
20050514                    13
20050832                    13
20051012                    23
20051020                    13
20060050                    13
20060240                    26

7 ligne(s) sélectionnée(s).
```

Ici on a sélectionné le ET_NUMERO des fiches études. Chaque fiche étude est caractérisée par ce numéro unique qui lui est propre.

Maintenant voici un exemple de sélection d'une étude ayant un nombre de produits nul :

```
SQL> (SELECT ET_NUMERO FROM ETUDES
  2  WHERE ETUDES.ET_ARCHIVE IS NULL)
  3  MINUS
  4  (SELECT PETUDES.ET_NUMERO FROM PETUDES, ETUDES
  5  WHERE PETUDES.ET_NUMERO = ETUDES.ET_NUMERO
  6  AND ETUDES.ET_ARCHIVE IS NULL
  7  GROUP BY PETUDES.ET_NUMERO
  8  HAVING COUNT(*) BETWEEN 1 AND 100);

ET_NUMERO
---------
20040018
20040036
 [...]
DV060263
DV060264

593 ligne(s) sélectionnée(s).
```

Cette recherche a été très utile dans la suite du développement. (cf : jeu de test). Elle a permis de prévoir les différentes tailles à affecter aux tableaux et de prévoir les messages d'erreurs quand les quantités (produits, conditions de test …) sont nulles.

4.8.3. L'affichage des données

Dans un premier temps, le développement de l'application a consisté a pouvoir uniquement affiché les données des fiches études. Les formulaires ayant auparavant été créés, nous avons codé les pages en PHP pour afficher les différentes valeurs de chaque champ. Nous nous sommes basé sur l'analyse faîte précédemment pour le codage des pages.

Nous avons tout d'abord créé chaque classe PHP correspondant à une table de la base de données. C'est à dire des classes qui forment un moule pour créer un objet. Par exemple la classe Etudes.php correspond à la table Etudes de la base de données et permet de créer une Etude.

Nous avons ensuite créé une classe FunctionDBManager. Cette classe permet de gérer la connexion et la déconnexion à la base de données mais surtout cette classe contient toutes les méthodes pour effectuer des requêtes sur la base de données.

Par exemple, voici la méthode qui retourne un Tiers à partir de son code d'identification (code unique pour chaque Tiers) :

```
/**
 * Retourne un tiers (objet TIERS) à partir
 * de son code tiers (TI_CODE).
 */
 public function get_tiers($code) {
 // Préparation de la requête
    $query = "SELECT * FROM Tiers
              WHERE TI_CODE = '".$code."'";
    $results = $this->dbm->select($query);

    $mon_tier = new Tiers($results, 0);

    return $mon_tier;
 }
```

Finalement, nous avons créé la classe Formateur. Cette classe permet de formater les résultats des requêtes de FunctionDBManager afin de leur donner un aspect graphique. (Au format HTML).

Par exemple, la méthode suivante :

```
/**
 * Permet d'afficher la liste des produits d'une étude
 * à partir du numéro de la fiche courante.
 * 1 ligne : 'Code Produit'
 */
 public function get_liste_pEtudes($numFiche, $numProd) {

    // code (pas important ici)

 }
```

Permet d'afficher le tableau suivant :

Produits

Code produit				
ESABI043	ESUC050S	ESUC051S	ESUC057S	ESUC058S

Tableau des produits d'une étude

4.8.4. La modification des données

Dans un deuxième temps, le développement de l'application a consisté à inclure la possibilité d'enregistrer les informations saisies dans les formulaires. La modification des données a été plus rapide que l'affichage des données. En effet nous connaissions les valeurs de chaque champs et nous avons pu réutilisé une partie du code réalisé pour l'affichage des données.
Nous avons créé des méthodes d'enregistrement dans la classe FunctionDBManager.

Voici par exemple la méthode d'enregistrement des conditions d'un test :

```
/**
 * Permet d'enregistrer les conditions de test
 */
public function update_tests_cond($test){

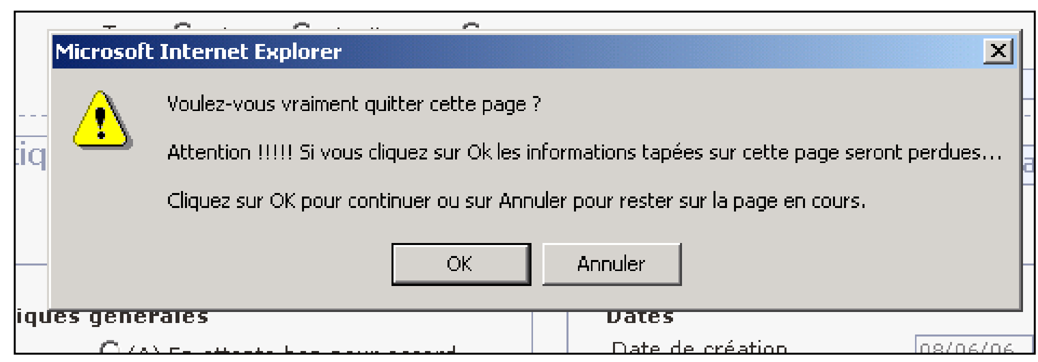
   $query = "UPDATE TESTS
            SET
            TO_DATEPREVUE = '".$test->get_to_dateprevue()."',
            TO_DATEREELLE = '".$test->get_to_datereelle()."'
            WHERE
            ET_NUMERO = '".$test->get_et_numero()."' AND
            TO_CPT = '".$test->get_to_cpt()."'";

            // On exécute la requête
            $this->dbm->update($query);

}
```

Pour enregistrer ses modifications, l'utilisateur doit cliquer sur le bouton « Enregistrer » situé en bas à droite de chaque page.

Si l'utilisateur change de page sans enregistrer ses modifications, ses données sont perdues. Nous avons donc mis en place un système de « popup » en *Javascript*.
Le *Javascript* est un langage de programmation, qui s'exécute du coté client sur le navigateur de la machine contrairement au PHP qui s'exécute du coté serveur.



Le popup de Javascript du changement de page

4.8.5. La gestion des collaborateurs

- **Le choix de la solution**

Pour gérer les comptes utilisateurs, le logiciel *SIGES* utilise une table de la base de données : la table Collaborateurs.

```
SQL> DESCRIBE COLLABORATEURS
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ------------
 CO_CODE                                   NOT NULL VARCHAR2(3)
 GR_CODE                                   NOT NULL VARCHAR2(10)
 CO_NOM                                    NOT NULL VARCHAR2(20)
 CO_PRENOM                                          VARCHAR2(20)
 CO_PASSWORD                               NOT NULL VARCHAR2(50)
 CO_SUPERUSER                                       VARCHAR2(1)
 CO_ACTIF                                  NOT NULL NUMBER(1)
```

Cette table définie les utilisateurs à l'aide d'un code (CO_CODE) qui correspond à leur login pour accéder au logiciel. Ce login est associé à un mot de passe (CO_PASSWORD). Ce mot de passe est crypté par le logiciel *SIGES*. Comme on ne connaît pas l'algorithme utilisé pour le cryptage de ce mot de passe, nous avons envisagé une nouvelle solution stocké les données des collaborateurs. Nous avons ainsi pensé à stocker les collaborateurs dans un fichier au format XML.

Qu'est ce qu'est le XML ? :

Le format XML est une évolution du langage SGML[1] permettant aux concepteurs de documents HTML de définir leurs propres marqueurs, dans le but de personnaliser la structure des données qu'ils comptent présenter.

Le XML est une recommandation du W3C[2]. L'objectif initial de XML était de faciliter le partage de textes et d'informations structurées, par exemple au travers de l'Internet, en séparant le contenu (les données) du contenant (la présentation des données).

[1]Standard Generalized Markup Language : langage général de marquage. Langage normalisé permettant de décrire les relations entre le contenu d'un document informatique et sa structure.

[2](World Wide Web Consortium)
Consortium industriel international fondé en 1994 pour développer des protocoles communs pour l'évolution du web.

- **La mise en place de la solution**

Nous avons créé un fichier qui contient toutes les informations de tous les utilisateurs de l'application. Il est placé dans le dossier *conf* et se nomme *collaborateurs.conf*.

Ainsi, si l'administrateur veut créer un nouvel utilisateur, le nom du nouveau collaborateur récupéré dans la table Collaborateurs de la base de données. Ensuite c'est ce fichier qui sera directement modifié. De même, si un utilisateur modifie ses paramètres de configuration, c'est ce fichier qui sera modifié.

Pour la gestion du fichier au format XML en PHP, nous avons utilisé un parseur DOM. Après différentes recherches sur Internet et sur le php_manual_fr, nous avons réussi à implémenter ce parseur. Nous avons quand même eu un problème. Au début nous n'arrivions qu'a enregistré un utilisateur a la fois. Après différents essais nous avons trouvé une solution pour enregistrer plusieurs collaborateurs.

- **Exemple de fichier collaborateurs.xml**

Voici la constitution du fichier lorsque l'application a 2 utilisateurs :


```
- <coll>
  - <collaborateur>
      <co_code>ADM</co_code>
      <mot_de_passe>63a9f0ea7bb98050796b649e85481845</mot_de_passe>
      <ti_code>SQUIMPER</ti_code>
      <nb_fiches_par_page>20</nb_fiches_par_page>
      <modele>modele 1</modele>
    </collaborateur>
  - <collaborateur>
      <co_code>VA</co_code>
      <mot_de_passe>aa36dc6e81e2ac7ad03e12fedcb6a2c0</mot_de_passe>
      <ti_code>STOULOUS</ti_code>
      <nb_fiches_par_page>40</nb_fiches_par_page>
      <modele>modele 3</modele>
    </collaborateur>
  </coll>
```


Aperçu du fichier collaborateurs.xml

Le premier collaborateur a comme :

- Login : ADM

Ce login est situé entre les balises <*co_code*></*co_code*>
Il est associé a un nom et un prénom, ici ADM est l'administrateur de l'application.

- Mot de passe : root

Le mot de passe est situé entre les balises <*mot_de_passe*><*mot_de_passe*>
Ici il est crypté au format *md5* pour sécuriser les comptes utilisateurs. Le *md5* est un algorithme de cryptage de PHP, ce type de cryptage produit des mots de passes haché, c'est-à-dire que les mots de passes sont indécryptables (on peut les crypter mais pas les décrypter).

- Lieu de test : SQUIMPER

Le lieu de test est situé entre les balises <*ti_code*></*ti_code*>
Il est choisi lors de la création du collaborateur. Différents lieux de tests sont possibles comme SPARIS, SCLERMONT …

- Nombre de fiches par page : 20

Le nombre de fiches par pages est situé entre les balises <*nb_fiches_par_page*></*nb_fiches_par_pages*>
C'est le nombre de fiches par page affichées dans la page de la liste des fiches études.

- Modèle : modele 1

Le modèle est situé entre les balises <*modele*></*modele*>
C'est le modèle de tableau affiché dans la page de la liste des fiches études.

4.8.6. La gestion de la connexion

- **Le choix de la solution**

Pour gérer la connexion, nous avons hésité entre utilisé les cookies ou les sessions en PHP. Les cookies sont des fichiers stockés par le navigateur Web sur la machine de l'utilisateur. Nous n'avons pas réussi à utilisé les cookies par conséquent nous avons donc utilisé les sessions en PHP.

- **La mise en place de la solution**

Le principe est le suivant :
Lorsqu'un utilisateur veut se connecter, il tape son login et son mot de passe. Nous récupérons le login et le mot de passe entrés. Ensuite, nous vérifions à l'aide du fichier *collaborateurs.xml* (cf : 4.7.5) si le mot de passe entré correspond bien au login. Pour cela, il faut crypter le mot de passe que l'utilisateur à entré au format md5 et le comparer au mot de passe crypté du fichier XML. Si ces mot de passe sons identique l'utilisateur se connecte. Cela créé une session PHP, c'est à dire un fichier contenant les informations de l'utilisateur qui est stocké dans le dossier session. Lorsque l'utilisateur n'est pas connecté, il n'a accès à aucune fonction de l'application. Lorsqu'il se connecte, le menu apparaît lui permettant de réaliser différentes actions.

Exemple :
L'utilisateur ADM entre le mot de passe 'root' pour se connecter.
On crypte le mot de passe au format md5, 'root' devient : '5c70bc1ae70578da1200f9dcda1b8f62'.
On compare ce mot de passe avec le mot de passe de l'utilisateur ADM stocké dans le fichier *collaborateurs.conf*.
Si ces mots de passe sons identique l'utilisateur se connecte.

Voici la bannière avant la connexion, le visiteur n'a accès a aucune fonctions de l'application :



La bannière avant la connexion

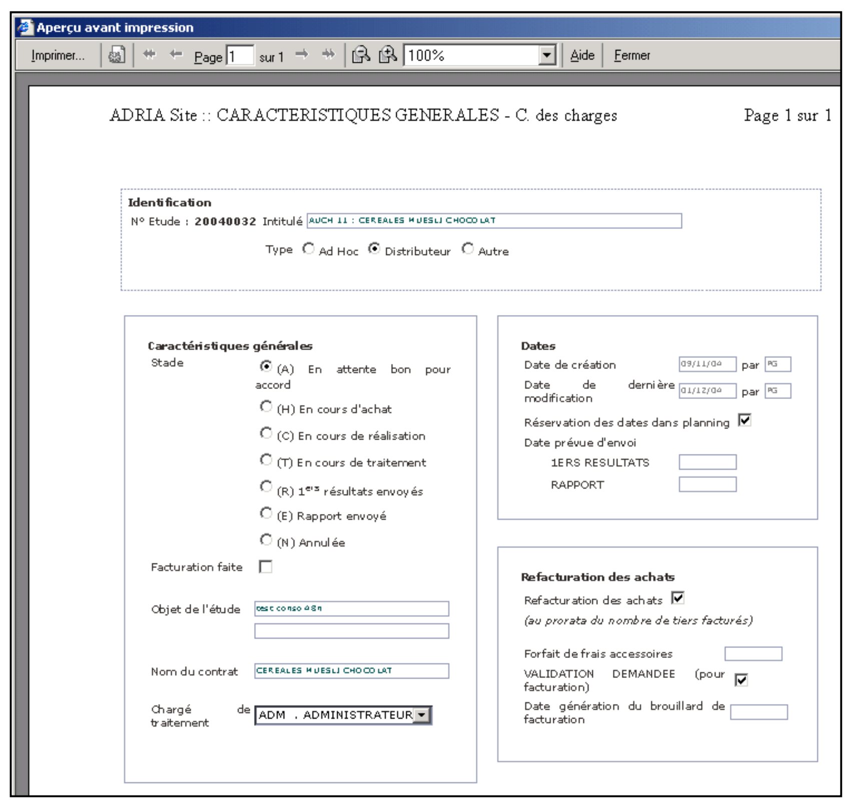
Une fois que l'utilisateur s'est connecté, un menu apparaît montrant les différentes fonctions de l'application :



La bannière après la connexion

4.8.7. L'impression des pages

L'impression des pages a posé un problème. Les utilisateurs de *SIGES* voulaient pouvoir imprimer des pages ressemblant fortement aux pages générés par le logiciel *SIGES*. Malheureusement cela aurait été beaucoup trop long. Pour l'impression nous nous sommes basés sur les formulaires déjà existant auxquels nous avons modifié les feuilles de styles (CSS) pour avoir un rendu plus propice à l'impression.



Aperçu de l'impression des caractéristiques sur l'application Web

Dans le cas des pages contenant des tableaux et des sous tableaux, nous avons créer un lien permettant de « formater la page pour l'impression ». C'est à dire que ce lien parcoure l'arborescence de tout les tableaux, pour les affichés sur différentes pages.

4.9. Plan de test de conformité

4.9.1. Les jeux de test

Afin de ne pas avoir (ou peu) de bugs dans l'application, j'ai mis en place un système de jeux de test. Ces jeux de test permettent de tester tout les scénario sur lesquels un utilisateur peut tomber. Ces tests ce sont bien comportés et on permis de ne pas oublié des bugs majeurs.

Voici un exemple de ces jeux de tests :

Les pages des onglets principaux			
Action	**Etat**	**Réaction**	**Validation**
Clique sur l'onglet Produits	L'étude ne contient pas de produits de l'étude	Affiche un message montrant qu'il n'y a aucun produit associé à cette étude	OK
	L'étude contient un nombre important de produits de l'étude	Affichage du tableau des produits de l'étude sans problèmes	OK
	Le produit de l'étude ne possède pas de souscripteur	Affiche un message montrant qu'il n'y a aucun souscripteur associé à ce produit de l'étude	OK
	Le produit de l'étude possède 1 souscripteur	Affichage du souscripteur dans un tableau	OK

Ce jeu de test permet de tester tous les cas possibles de la page produits de l'étude. Ainsi, l'utilisateur ne tombera pas sur une erreur ou sur un bug en visitant cette page.

4.9.2. Le test des utilisateurs sur la base de test

La seconde partie du test de l'application a consisté à fournir l'application aux personnes qui l'utiliseraient dans le futur, c'est-à-dire : Jessy CAVALAZZI, Murielle GALESNE et Alexis MARTY. Cette partie du test s'est effectué sur une base de test et non sur la vraie base de travail. Avant de leur fournir créer un compte pour l'application, une démonstration du logiciel à été faîte en compagnie de Jessy et Murielle. Alexis qui travaille au sein de l'ADRIA SENSO à Toulouse était au téléphone a effectué les mêmes manipulations sur l'application. Des premières remarques ont été dites, ce qui a permis d'apporter des améliorations à l'application (duplication des dates de l'onglet sous test conso par exemple).

Ensuite ces personnes ont testé l'application durant 2 jours et on noté toutes leur remarque.

4.9.3. Le déploiement de l'application sur le serveur de l'ADRIA SENSO

La dernière étape a été de porter l'application sur le serveur de l'ADRIA SENSO et de la lier avec la base de données utilisée actuellement par les utilisateurs de *SIGES*. Cette étape n'a pas posé de problème car le dossier de développement était bien rédigé et contenait toutes les informations nécessaires au bon déploiement de la base. L'application est devenue beaucoup plus rapide avec le serveur de l'ADRIA SENSO. L'application est maintenant accessible par l'adresse « *http://srv_senso_01/* »

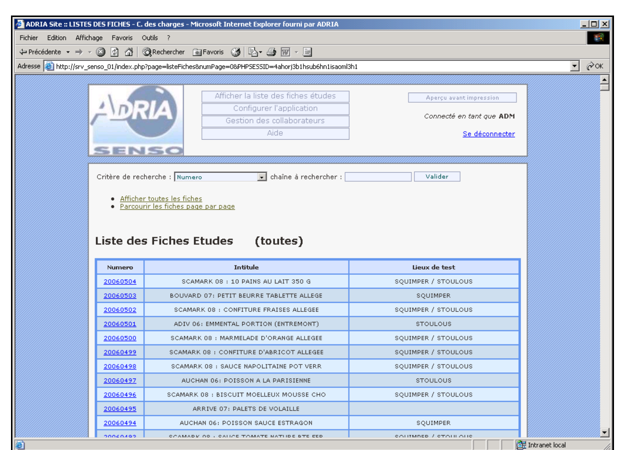
Par exemple, il fallait une vingtaine de secondes pour afficher la liste de toutes les études. Maintenant, grâce au serveur de l'ADRIA SENSO, il ne faut plus que 6 à 7 secondes. (cf : Test validation performance)

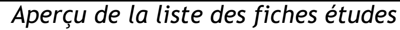
5. RESULTAT

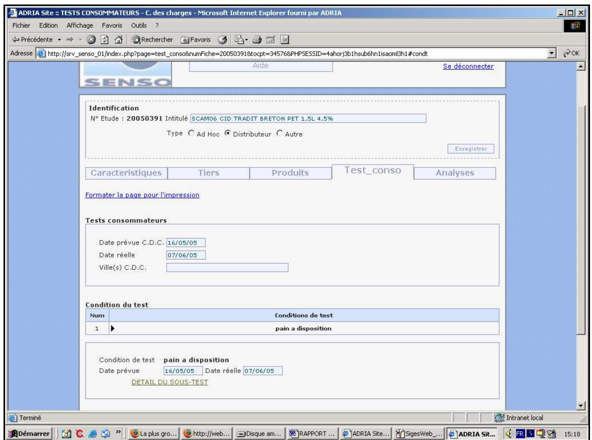
Le résultat final est satisfaisant. Le cahier des charges a été respecté, toutes les fonctionnalités prévues ont été implémentées. De plus, l'application fonctionne rapidement et sans bugs connus à ce jour. Concernant le planning prévisionnel, il a en partie été respecté, nous avons mis une semaine de plus à codé l'application, ce qui nous a amené à 2 semaines de tests au lieu des 3 initialement prévues. C'est surtout la mise en place de la base de données de test qui nous a retardé et bloqué au début du stage. Après cela, le codage s'est déroulé sans problèmes majeurs.

L'application contient environ 16000 lignes de codes et à 130 pages d'annexes associées. (cf : 8. Annexes)

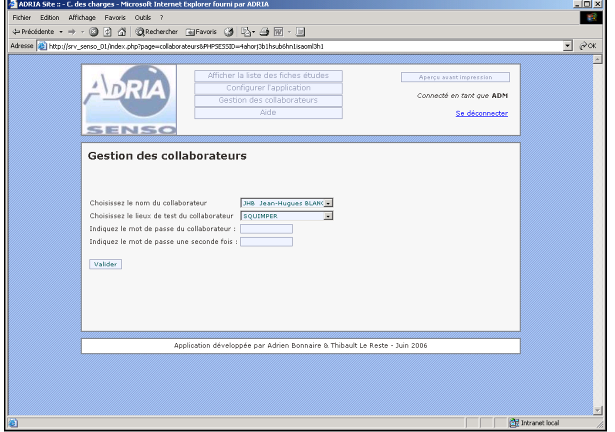
Voici maintenant quelques aperçus des pages de l'application :



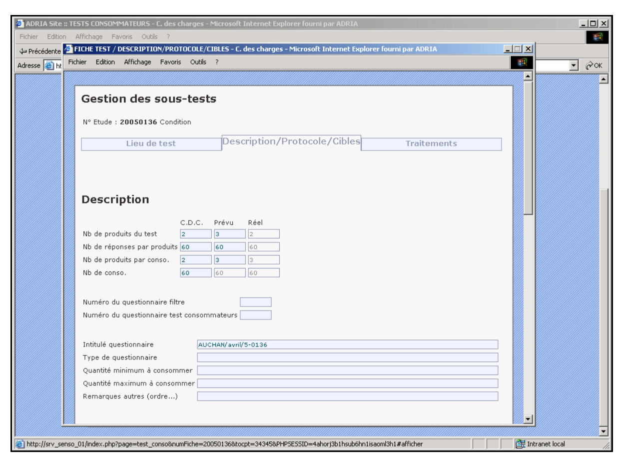

Aperçu de la liste des fiches études



Aperçu de la page des tests consommateurs



Aperçu de la page de gestion des collaborateurs



Aperçu de la page de gestion des sous-tests consommateurs

6. CONCLUSION

- **Bilan professionnel :**

Ce stage de fin de DUT m'a permis de travaillé sur un grand projet de conception d'une application Web. Ce fut la première expérience professionnelle qui s'est bien déroulée au vu des résultats obtenus. Ce stage m'a permit d'apprendre à mettre en place une base de données de test. Il m'a aussi permis d'utiliser mes connaissances pour installer de PHP5 et d'un serveur Apache. Du point de vue du codage, j'ai augmenté mon niveau en PHP5. Ce projet développé de A à Z, m'a permit d'appliquer et de confirmer mes connaissances de l'analyse d'un projet informatique. Le projet, fonctionnel, constitue maintenant un nouvel outil pour l'entreprise.

- **Bilan personnel :**

J'ai passé un bon stage en compagnie de mon collègue de travail. Le travail un binôme s'est bien déroulé et a été un moteur de travail. En effet, l'entraide, les conseils entre nous deux, a été permit un développement rapide et efficace. Nous avons su gérer le temps à notre disposition. Je suis satisfait du résultat obtenu. En effet, le projet qui devait être seulement un test au début, sera réellement utilisé par l'Entreprise. Néanmoins, j'aurais apprécié travaillé au sein d'une équipe informatique, pour connaître les méthodes de travail utilisées.

- **Perspectives :**

De nombreuses améliorations peuvent être développées sur l'application existante. Par exemple, on peut imaginé un système d'impression plus poussée qui ressemblerais à l'impression de *SIGES*. On pourrais aussi recréer toute la base de données, c'est à dire toute les tables et leur relations entre elles pour avoir un système optimisé et correspondant mieux aux besoins des utilisateurs.

7. BIBLOGRAPHIE

Documents papier :

- Application LEONI V2 Manuel Utilisateur
- ADIV MARKETING Document d'analyse Gestion des Etudes
- Cours de base de données de l'IUT

Documents informatiques :

- PHP5-manuel fr
- Codes source de LEONI / SIGES

Site Internet:

- forum.hardware.fr
- www.alsacreations.com
- www.developpez.com
- www.wikipedia.fr

8. ANNEXES

Voici la liste des annexes de ce rapport de stage :

- **Le document d'étude des langages serveurs**
  Ce document contient une analyse sur les langages serveurs. Il a été utilisé au début du stage pour définir quel langage utilisé. Finalement c'est le PHP5 qui a été choisi.

- **Le manuel d'installation**
  Ce document contient les différentes étapes pour monter une base de Test. Il contient aussi la démarche pour installer et configurer un serveur Apache avec PHP5.

- **Le dossier de spécification**
  Ce document contient toute l'analyse effectuée pour mettre au point l'application.

- **Le manuel d'utilisation**
  Ce document est le manuel d'utilisation de l'application.

- **Le document de résultat des tests de fonctionnalités et de performance**
  Ce document contient les jeux de tests de l'application accompagnés des résultats de ces tests. Il contient aussi des tests de performance de l'application.

Résumé

ADRIA SENSO est une société spécialisée dans l'évaluation et la caractérisation sensorielle. Elle réalise des prestations de tests consommateurs, de profils sensoriels et de la formation aux techniques sensorielles. Pour réaliser ses tests consommateurs et analyses, le groupe ADRIA SENSO utilise des fiches études. Celles-ci sont gérées par le logiciel SIGES. Le but de ce stage est de concevoir une application Web de type intranet ayant les même fonctionnalités que SIGES, ceci afin de gagner en rapidité, ergonomie et coûts. Cette application est liée à une base de données Oracle et est développée en PHP5. Le travail a été effectué par un binôme de deux étudiant de l'IUT informatique de Vannes et s'est déroulé pendant une période de 10 semaines.

Mots clés

ADRIA SENSO, SIGES, PHP5, Oracle, application Web, test consommateurs, analyses, fiches études

Abstract

ADRIA SENSO is a company specialized in the evaluation and the sensory characterization. It carries out services of consuming tests, profiles sensory and formation with the sensory techniques. To carry out its consuming tests and analyses, group ADRIA SENSO uses cards studies. Those are managed by software SIGES. The goal of this training course is to conceive a Intranet Web application having same the functionalities as SIGES, this in order to gain in speed, ergonomics and costs. This application is related to a data base Oracle and is developed in PHP5. Work was carried out by a binomial of two student of the Vannes IT IUT and proceeded for 10 weeks period.

KeyWords

ADRIA SENSO , SIGES, PHP5, Oracle, Web application, consuming tests, analyses, cards studies